6/9/1 (Item 1 from file: 348)
DIALOG(R)File 348:EUROPEAN PATENTS

01026897
SOFTWARE SYSTEM GENERATION
SOFTWARESYSTEMERZEUGUNG
PRODUCTION DE SYSTEME LOGICIEL
PATENT ASSIGNEE:
BRITISH TELECOMMUNICATIONS public limited company, (846100), 81 Newgate Street, London EC1A 7AJ, (GB), (Proprietor designated states: all)
INVENTOR:
LEE, Lyndon Chi-Hang, 17 Pampas Close,Colchester, Essex CO4 4ST, (GB)
NWANA, Hyacinth, Sama, 17 Tallboys Close,Grange Farm,Kesgrave, Ipswich,Suffolk IP5 2YF, (GB)
NDUMU, Divine, Tamajong, 11 Hilton Road,Martlesham Heath, Ipswich,Suffolk IP5 7RL, (GB)
LEGAL REPRESENTATIVE:
Dutton, Erica L. G. (63161), BT Group Legal Services, Intellectual Property Department, 8th Floor, Holborn Centre 120 Holborn, London EC1N 2TE, (GB)
PATENT (CC, No, Kind, Date): EP 996886 A1 000503 (Basic)
EP 996886 B1 021009
WO 99005593 990204
APPLICATION (CC, No, Date): EP 98936513 980727; WO 98GB2241 980727
PRIORITY (CC, No, Date): EP 97305600 970725
DESIGNATED STATES: BE; CH; DE; ES; FR; GB; IT; LI; NL
INTERNATIONAL PATENT CLASS: G06F-009/44
CITED PATENTS (WO A): XP 2046843 ; XP 4018198 ; XP 446368
CITED REFERENCES (EP B):
WONG D ET AL: "Concordia: an infrastructure for collaborating mobile agents" MOBILE AGENTS. FIRST INTERNATIONAL WORKSHOP, MA '97 PROCEEDINGS, MOBILE AGENTS. FIRST INTERNATIONAL WORKSHOP, MA '97.
PROCEEDINGS, BERLIN, GERMANY, 7-8 APRIL 1997, pages 86-97, XP002046843
ISBN 3-540-62803-7, 1997, BERLIN, GERMANY, SPRINGER-VERLAG, GERMANY
CIANCARINI P ET AL: "PageSpace: An architecture to coordinate distributed applications on the web" COMPUTER NETWORKS AND ISDN SYSTEMS, vol. 28, no. 11, May 1996, page 941-952 XP004018198
HUANG Y -M ET AL: "DESIGNING AN AGENT SYNTHESIS SYSTEM FOR CROSS-RPC COMMUNICATION" IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, vol. 20, no. 3, 1 March 1994, pages 188-198, XP000446368;

CITED REFERENCES (WO A):
WONG D ET AL: "Concordia: an infrastructure for collaborating mobile agents" MOBILE AGENTS. FIRST INTERNATIONAL WORKSHOP, MA '97 PROCEEDINGS, MOBILE AGENTS. FIRST INTERNATIONAL WORKSHOP, MA '97. PROCEEDINGS, BERLIN, GERMANY, 7-8 APRIL 1997, pages 86-97, XP002046843 ISBN 3-540-62803-7, 1997, BERLIN, GERMANY, SPRINGER-VERLAG, GERMANY
CIANCARINI P ET AL: "PageSpace: An architecture to coordinate distributed applications on the web" COMPUTER NETWORKS AND ISDN SYSTEMS, vol. 28, no. 11, May 1996, page 941-952 XP004018198
HUANG Y -M ET AL: "DESIGNING AN AGENT SYNTHESIS SYSTEM FOR CROSS-RPC COMMUNICATION" IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, vol. 20, no. 3, 1 March 1994, pages 188-198, XP000446368;
NOTE:
No A-document published by EPO
LEGAL STATUS (Type, Pub Date, Kind, Text):

Application:	000503 A1 Published application with search report
Application:	990512 A1 International application (Art. 158(1))
Lapse:	040121 B1 Date of lapse of European Patent in a contracting state (Country, date): BE 20021009, CH 20021009, LI 20021009, ES 20030429, NL 20021009,
Oppn None:	031001 B1 No opposition filed: 20030710
Lapse:	030723 B1 Date of lapse of European Patent in a contracting state (Country, date): NL 20021009,
Examination:	020102 A1 Date of dispatch of the first examination report: 20011109
Examination:	000503 A1 Date of request for examination: 20000106
Grant:	021009 B1 Granted patent
Lapse:	030924 B1 Date of lapse of European Patent in a contracting state (Country, date): CH 20021009, LI 20021009, NL 20021009,
Lapse:	031217 B1 Date of lapse of European Patent in a contracting state (Country, date): BE 20021009, CH 20021009, LI 20021009, NL 20021009,

LANGUAGE (Publication,Procedural,Application): English; English; English
FULLTEXT AVAILABILITY:

Available Text	Language	Update	Word Count
CLAIMS B	(English)	200241	935
CLAIMS B	(German)	200241	913

CLAIMS B (French) 200241 1110
SPEC B (English) 200241 20617
Total word count - document A 0
Total word count - document B 23575
Total word count - documents A + B 23575

SPECIFICATION EP 996886 B1

The present invention relates to software system generation and finds an application for instance in building communications systems.

Software agent technology has developed over the past few years in several different fields. A software agent is a computer program which acts as an agent for an entity such as a user, a piece of equipment or a business. The software agent usually holds data in relation to the entity it represents, has a set of constraints or conditions to determine its behaviour and, most importantly, is provided with decision making software for making decisions on behalf of the entity within or as a result of the constraints and conditions. Agents are generally acting within a system and the decisions an agent makes can result in activity by the system. In control software systems, those decisions result in control activity, such as initiating connection set-up in a communications network controlled by the system.

An agent acting within a system will also generally hold data about the system so that it can operate in context.

In a distributed environment, many such agents may co-operate to coordinate and perform the control activities. Typically, such agents form an agent layer, with each agent interfacing with a number of external systems (the domain layer) which they control, monitor or manage, as shown in Figure 1.

An agent-based system can be very complex since the interactions between the agents, the decision-making processes of individual agents and the interactions between agents and the external systems they control need to be taken into account.

Different types of agent-based systems are described in many papers, such as those published in the proceedings of the First and Second International Conferences on the Practical Application of Intelligent Agents and Multi-Agent Technology. These are published by the Practical Application Company Ltd., Blackpool, Lancashire, in 1996 and 1997 respectively. A general comprehensive review of agent-based technology is given by Hyacinth S. Nwana, "Software Agents. An Overview" in the Knowledge Engineering Review journal, Vol. 11, No. 3, pages 205-244.

There are ways already known for building software agents. For instance, the following publications describe agent building arrangements:

1. IBM's Agent Building Environment (ABE) which is essentially a C + + class library (http://www.networking.ibm.com/iag/iagwatsn.htm). ABE is a tool-kit that facilitates the construction of agent-based applications or

helps add an agent to existing applications. This tool-kit applies to relatively trivial "interface" agents, or agents that work alone. For example, an agent here could be one that which monitors the value of stock in the financial markets and alerts its user (e.g. via paging) when the value falls below a certain threshold. ABE does not describe means for building multiple agent systems, nor do they describe means for building more than one type of agent.

2. MIT's SODABOT (http://www.ai.mit.edu/people/sodabot/sodabot.html), General Magic's Telescript and Odyssey (http://www.genmagic.com), and IBM's Aglets (http://www.trl.ibm.co.jp/aglets). These all provide other environments which facilitate the construction of "mobile" agents-based applications. However, they are also not much more than languages, comparable to the "Java" language developed by Sun Microsystems Inc., and do not provide specific advanced agent-building arrangements.

Hayes-Roth etal, in paper publication "Domain specific software architectures: distributed intelligent control and management", presented in Computer-Aided Control System Design 1992 pp. 117 - 128, presents an environment for designing and creating a generic multi-agent system (agents are termed controllers in Hayes-Roth etal). The architecture of the agents themselves is also described. The Hayes-Roth etal multi-agent systems are designed for control problems, specifically having application in manufacturing, vehicle management, robotics etc.

The agents that are created using the agent-creating environment are interrelated in a hierarchical manner; agents have specific beliefs and intentions, and they can communicate with a predefined group of agents as a function of their position in the hierarchy. There are set functionalities associated with each level in the hierarchy and, as a result, inter-agent interaction is fixed.

Perhaps more relevant to embodiments of the present invention is the agent building shell work done at the University of Toronto. This is described by Mihai Barbuceanu & Mark S. Fox in the paper "The Architecture of an Agent Building Shell", published in 1996 in Intelligent Agents II, Berlin by Springer-Verlag, 1037, 235-250 and edited by Wooldridge, M., Muller, J. & Tambe, M. This work describes an agent building shell "that provides several reusable layers of languages and services for building agent systems: coordination and communication languages, description logic based knowledge management, cooperative information distribution, organisation modelling and conflict management" (page 235). This work is still very much in progress and has not yet resulted in a practical embodiment with much effort having been expended on theoretical issues such as description logics, non-monotonic logics and extending KQML to derive the language "COOL". The work of the present invention differs markedly from this in that it provides a tool-kit for defining and generating real agent-based control software for real applications, and goes well beyond the general academic nature of the Toronto work.

A particular problem arises with "collaborative" agents. Collaborative agents are a group of agents which co-operate with one another to co-ordinate their activities in performing a particular task. Such co-operation may be necessary because know-how, resources or processing power may be distributed across the environment or across the agents in the system. The problem with collaborative agent systems is the need to co-ordinate their activities in a problem-and context-dependent manner.

An example of a collaborative agent system, used in this case in communications network management, is described in international patent application number WO95/15635, in the name of the present applicant.

According to a first aspect of the present invention, there is provided software building apparatus, for building a software system for use in control, monitoring and/or management of a process or apparatus, said apparatus comprising:

i) at least one software module,

ii) means for capturing data for loading to at least two copies of said module, the loaded modules each comprising a collaborative software agent for use in said software system,

iii) means for loading, or providing access to, at least two different collaboration strategies to at least one of said copies of said loaded module, for use by the respective collaborative software agent in use of the software system, and

iv) means for generating the software system comprising the collaborative software agents of ii) and iii).

As each loaded software module comprises a collaborative software agent, it will therefore comprise or have access to at least two collaboration or co-ordination strategies, expressed for instance as a rule or algorithm. The collaborative software agents together can then provide a multiple agent community for controlling, monitoring and/or managing the process.

Embodiments of the present invention can provide collaborative agent building environments with which system developers can define a set of agents to work together in a system, organise them in relation to one another in whatever manner they choose, imbue them with co-ordination abilities suitable for the problems the agents are being designated to tackle, support links from the agents to said process or apparatus they need to communicate with, to control or update for instance, and generate the software code for the agents.

It is not necessary that all the loaded software modules are the same. Indeed, usually at least some of them will hold different data sets because they represent different entities. Thus, the software building environment provides, or provides access to, more than one collaboration or co-ordination strategy. In use of the environment, the developer can then load these different collaboration or co-ordination strategies to at least one module for use in the system.

According to a second aspect of the present invention, there is

provided a software system for use in control, monitoring and/or management of a process or apparatus, wherein said system comprises at least two software modules, each module comprising data and/or process information which comprises:

(i) organisation data concerning an inter-module relationship; and

(ii) executable software providing at least two inter-module collaboration strategies, expressed for instance as a rule or algorithm;

wherein, in use, a module selects at least one of said at least two strategies for use in negotiating with another software module in relation to task allocation, said selection being determined at least in part by said organisation data.

It should be noted that there are several novel and innovative features of the embodiments of the present invention described below, not all of which are necessarily referred to above, and at least some of which have applicability independently of other aspects of said embodiments.

It should also be noted that, in a distributed environment, software modules in practice may not themselves comprise data or software, such as collaboration or co-ordination strategies as mentioned above. They may instead simply have access to them, for instance for loading at the relevant run-time. These arrangements should be taken as covered by the above.

An agent building system tool-kit known as the Collaborative Agent Building System ("CABS") will now be described, by way of example only, as an embodiment of the present invention, with reference to the accompanying drawings, in which:

Figure 1 shows an agent-based control system, built using CABS, as it interfaces with external hardware and/or software;

Figure 2 shows a schematic architecture for a software module constituting an agent for distributed control, monitoring and management of a system;

Figure 3 shows a CABS platform for building an agent as shown in Figure 2;

Figure 4 shows a layered model of an agent in terms of primary characterisation;

Figure 5 shows a flow chart of steps involved in designing an agent using the platform of Figure 3;

Figure 6 shows possible organisational relationships between software agents built using CABS;

Figure 7 shows data obtained using a debugging tool for debugging an agent-based control system built using CABS;

Figure 8 shows a scenario for debugging using the debugging tool for debugging a CABS agent system;

Figure 9 shows a commitment table for an agent according to Figure 2;

Figure 10 shows a debugging and visualisation system for use with the agent-based control system of Figure 1;

Figure 11 shows a flow chart of a co-ordination process for use

ontologies or date dictionaries to specify date items, this tool allows user to load an ontology database defining the ontology of their specific application.

5.5 The Statistics Tool

Referring to Figure 15, this tool allows a user to collate various statistics about a society of agents on a per agent basis as well as on a community basis. The statistics collected might include for instance, but are not limited to:

(a) the number of messages and their types sent by the agents over a time period;

(b) the number of messages sent by the agents in coordinating different jobs;

(c) the average loading of the agents, i.e. the proportion of time agents spend actually executing tasks; and

(d) the coordination versus execution time ratio, i.e. how much time is spent coordinating tasks as opposed to actually running them.

Statistics such as these are particularly useful in debugging or fine-tuning the performance of a society of agents. For example, using the control monitor tool and the statistics tool, a user is better able to answer questions such as "what organisational structuring, and distribution of task and coordination know-how best minimises the time spent coordinating jobs and maximises the time spent executing them (i.e. increasing the profitability of the society)". If the agents in the society learn from experience, the statistics tool becomes even more useful in assessing the performance of different learning strategies since certain statistics such as the average number of messages sent by the agent per job and/or the coordination versus execution time ratio can serve as performance metrics for measuring learning success.

The particular screen shown in Figure 15 shows the amount of traffic generated 1500 by the principal agents U, T, P, M, C in achieving the goal MakeComputer. As expected from the task decomposition 700 shown in Figure 13, Agent C sends out three achieve-messages while Agent P only throws out one. The reply messages correspond to negotiation and post-negotiation (task management) dialogues.

The tool can support different display formats, such as histogram, pie, line, or xy-scatter graph formats for the display of statistics. The choice of format is user-determinable, using toolbar buttons 1505, although there are pre-specified defaults formats for the various statistics.

Similarly to the society and report tools, the statistics tool supports database saving and video-style replay of agent statistics. For generality, only raw (as opposed to processed) agent data is saved onto database, thus, on playback, any relevant statistic can berecreated.

5.6 An Extended Example

7. Softwareaufbauvorrichtung nach Anspruch 6, bei der die Kollaborationsmaschine eine Kollaborationsstrategie bei Verwendung des Softwaresystems ablaufen lassen kann, indem sie einen ersten Knoten der zugeordneten Graph-Beschreibung wahlt, einen durch den Knoten identifizierten Prozes instanziiert und ablaufen last und einen Prozes, der durch einen dem Knoten zugeordneten Bogen identifiziert wird, instanziiert und ablaufen last, wodurch ein zweiter Knoten durchlaufen wird, und indem sie den Prozes wiederholt, bis das Ende der Graph-Beschreibung erreicht ist.
8. Softwareaufbauvorrichtung nach Anspruch 6, wenn abhangig von Anspruch 3, bei der die Kollaborationsmaschine die wenigstens zwei Kollaborationsstrategien ablaufen lassen kann, indem sie parallel einen Knoten aus jeder der Strategien wahlt, Prozesse, die durch alle gewahlten Knoten identifiziert werden, instanziiert und ablaufen last, dann von jeder der Strategien zu einem nachsten Knoten ubergeht und Prozesse, die durch alle diese nachsten Knoten identifiziert werden, instanziiert und ablaufen last und den Prozes wiederholt, bis das Ende jeder Graph-Beschreibung, die einer Kollaborationsstrategie zugeordnet ist, erreicht ist.
9. Softwaresystem fur die Verwendung bei der Steuerung, der Uberwachung und/oder dem Management eines Prozesses oder einer Vorrichtung, wobei das System wenigstens zwei Softwaremodule umfast, wovon jedes Daten und/oder Prozesinformationen enthalt, die umfassen:

i) Organisationsdaten, die die Beziehung zwischen den Modulen betreffen; und

ii) ausfuhrbare Software, die wenigstens zwei Kollaborationsstrategien zwischen den Modulen schafft; wobei im Gebrauch ein Modul (300) wenigstens eine der wenigstens zwei Strategien auswahlt, um sie bei Verhandlungen mit einem weiteren Softwaremodul in bezug auf eine Aufgabenzuweisung zu verwenden, wobei die Auswahl wenigstens teilweise durch die Organisationsdaten bestimmt ist.

10. Softwaresystem nach Anspruch 9, bei dem wenigstens zwei der Softwaremodule unterschiedliche Daten halten, die durch unterschiedliche Entitaten bestimmt sind, die durch die Module reprasentiert werden.
11. Softwaresystem nach Anspruch 9 oder Anspruch 10, bei dem wenigstens ein Modul, das ausfuhrbare Software enthalt, die wenigstens zwei unterschiedliche Kollaborationsstrategien schafft, wahrend derselben Zeitperiode gemas mehr als einer Kollaborationsstrategie arbeiten kann, so das es wenigstens gemas zweier unterschiedlicher Strategien effektiv parallel arbeitet.
12. Softwaresystem nach einem der Anspruche 9 bis 11, bei dem jedes Modul eine Kollaborationsmaschine (210) enthalt, die im Gebrauch des Systems eine Kollaborationsstrategie ablaufen last.
13. Softwaresystem nach einem der Anspruche 9 bis 12, bei dem eine Kollaborationsstrategie eine Graph-Beschreibung enthalt.

14. Softwaresystem nach Anspruch 13, bei dem die Graph-Beschreibung eine Menge von Zustanden, die durch Knoten identifiziert sind, sowie eine Menge von Bogen, die zwischen spezifizierten Zustanden verlaufen, beschreibt, wobei jeder Knoten und jeder Bogen wenigstens einen Prozes identifiziert.
15. Softwaresystem nach Anspruch 14, bei dem wenigstens ein Bogen eine Graph-Beschreibung identifiziert.
16. Softwaresystem nach Anspruch 15, bei dem die Kollaborationsmaschine eine Kollaborationsstrategie im Gebrauch des Softwaresystems ablaufen lassen kann, indem sie einen ersten Knoten der zugeordneten Graph-Beschreibung auswahlt, einen durch den Knoten identifizierten Prozes instanziiert und ablaufen last, einen Prozes, der durch einen dem Knoten zugeordneten Bogen identifiziert wird, instanziiert und ablaufen last, um zu einem zweiten Knoten uberzugehen, und den Prozes wiederholt, bis das Ende der Graph-Beschreibung erreicht ist.
17. Ausnahmehandhabungseinrichtung fur die Verwendung in einer Vorrichtung nach einem der Anspruche 1 bis 8, bei der die aufgenommenen Daten Aufgabendefinitionsdaten enthalten, die Betriebsmittel und andere Variablen von Aufgaben definieren, die zwischen mehreren geladenen Kopien von Softwaremodulen fur die Verwendung in dem Softwaresystem verteilt werden sollen, wobei die Aufgabendefinitionen eine oder mehrere Beschrankungen in bezug auf eine oder mehrere Variablen umfassen und die Ausnahmehandhabungseinrichtung so beschaffen ist, das sie eine Ausnahme ausgibt, wenn eine Variable wahrend der Verwendung des Systems mit einer relevanten Beschrankung in Konflikt gerat.
18. Ausnahmehandhabungseinrichtung fur die Verwendung in der Vorrichtung nach einem der Anspruche 1 bis 8, bei der die aufgenommenen Daten Zeitplandaten fur Aufgaben enthalten, die Betriebsmittel- und Zeitbeschrankungen in bezug auf eine Aufgabe definieren, und die Ausnahmehandhabungseinrichtung so beschaffen ist, das sie eine Ausnahme ausgibt, wenn die Betriebsmittel- und Zeitbeschrankungen durch Zuweisen von Aufgaben zwischen den Softwaremodulen nicht erfullt werden konnen.

CLAIMS EP 996886 B1

1. Dispositif de construction de logiciel destine a construire un systeme de logiciel pour une utilisation dans la commande, la surveillance et/ou la gestion d'un procede ou d'un dispositif, ledit dispositif comprenant :

i) au moins un module de logiciel (300),

ii) un moyen (305) destine a acquerir des donnees en vue d'un chargement dans au moins deux copies dudit module, les modules charges comprenant chacun un agent de logiciel collaborateur pour une utilisation dans ledit systeme de logiciel,

iii) un moyen (385) destine a generer un systeme de logiciel,

caracterise par

iv) un moyen (210, 255) destine a charger, ou a fournir un acces a au moins deux strategies de collaboration differentes, vers au moins une desdites copies dudit module charge, pour une utilisation par l'agent de logiciel collaborateur respectif dans l'utilisation d'un systeme de logiciel, et et en ce que le moyen de generation est agence pour generer un systeme de logiciel comprenant les agents de logiciel collaborateurs de iii) et iv).

2. Dispositif de construction de logiciel selon la revendication 1, dans lequel au moins deux desdits modules de logiciels charges contiennent des ensembles de donnees et/ou de logiciels differents, les ensembles de donnees et/ou de logiciels etant determines par les entites representees par lesdits modules.
3. Dispositif de construction de logiciel selon la revendication 1 ou la revendication 2, dans lequel le au moins un module de logiciel collaborateur comprenant au moins deux strategies differentes, peut fonctionner selon plus d'une strategie de collaboration pendant la meme duree, de sorte qu'il fonctionne selon au moins deux strategies differentes effectivement en parallele.
4. Dispositif de construction de logiciel selon l'une quelconque des revendications precedentes, dans lequel le module de logiciel comprend un moteur de collaboration (210) devant etre utilise par un agent de logiciel collaborateur, le moteur de collaboration etant agence pour executer une strategie de collaboration.
5. Dispositif de construction de logiciel selon la revendication 4, dans lequel une strategie de collaboration comprend une description de graphe.
6. Dispositif de construction de logiciel selon la revendication 5, dans lequel la description de graphe decrit un ensemble d'etats, identifie par des noeuds, et un ensemble pour des arcs a parcourir entre les etats specifies, chaque noeud et chaque arc identifiant au moins un procede.
7. Dispositif de construction de logiciel selon la revendication 6, dans lequel le moteur de collaboration est concu pour effectuer une strategie de collaboration, par l'utilisation du systeme de logiciel, en selectionnant un premier noeud de la description de graphe associee, en instanciant et en realisant un procede identifie par le noeud, en instanciant et en realisant un procede identifie par un arc associe a un noeud, en passant ainsi a un second noeud, et en repetant le procede jusqu'a la fin de la description de graphe.
8. Dispositif de construction de logiciel selon la revendication 6 lorsqu'elle depend de la revendication 3, dans lequel le moteur de collaboration est concu pour executer les au moins deux strategies de collaboration en selectionnant en parallele, un noeud dans chacune des strategies, en instanciant et en realisant les procedes identifies par tous les noeuds selectionnes, et en passant au noeud

suivant depuis chacune des strategies et en instanciant ainsi qu'en realisant des procedes identifies par tous les noeuds suivants, et en repetant le procede jusqu'a la fin de chaque description de graphe associee a la strategie de collaboration.

9. Systeme de logiciel destine a etre utilise pour la commande, la surveillance et/ou la gestion d'un procede ou d'un dispositif, dans lequel ledit systeme comprend au moins deux modules de logiciels, chaque module comprenant des informations de donnees et/ou de traitement qui comprennent :

(i) des donnees d'organisation (215) concernant une relation inter-modules, et

(ii) un logiciel executable fournissant au moins deux strategies de collaboration inter-modules, dans lequel, en utilisation, un module (300) selectionne au moins une desdites au moins deux strategies pour une utilisation dans une negociation avec un autre module de logiciel en relation a une allocation de tache, ladite selection etant determinee au moins en partie par lesdites donnees d'organisation.

10. Systeme de logiciel selon la revendication 9, dans lequel au moins deux desdits modules de logiciels comprennent des donnees differentes, determine par des entites differentes representees par lesdits modules.

11. Systeme de logiciel selon la revendication 9 ou la revendication 10, dans lequel au moins un module, comprenant un logiciel executable fournissant au moins deux strategies de collaboration differentes, est capable de fonctionner selon plus d'une strategie de collaboration pendant le meme intervalle de temps, de sorte qu'il fonctionne selon au moins deux strategies differentes effectivement en parallele.

12. Systeme de logiciel selon l'une quelconque des revendications 9 a 11, dans lequel chaque module comprend un moteur de collaboration (210) destine a executer une strategie de collaboration en utilisant le systeme.

13. Systeme de logiciel selon l'une quelconque des revendications 9 a 12, dans lequel une strategie de collaboration comprend une description de graphe.

14. Systeme de logiciel selon la revendication 13, dans lequel la description de graphe decrit un ensemble d'etats, identifie par des noeuds, et un ensemble pour des arcs a parcourir entre les etats specifies, chaque noeud et chaque arc identifiant au moins un procede.

15. Systeme de logiciel selon la revendication 14, dans lequel au moins un arc identifie une description de graphe.

16. Systeme de logiciel selon la revendication 15, dans lequel le moteur de collaboration est concu pour executer une strategie de collaboration, en utilisant le systeme de logiciel, en selectionnant un premier noeud de la description de graphe associee, en instanciant

et en executant un procede identifie par le noeud, en instanciant et en executant un procede identifie par un arc associe a un noeud, en passant ainsi a un second noeud, et en repetant le procede jusqu'a la fin de la description de graphe.

17. Gestionnaire d'exception pour une utilisation dans un dispositif selon l'une quelconque des revendications 1 a 8, dans lequel les donnees acquises comprennent des donnees de definition de tache, definissant des ressources et d'autres variables de taches devant etre reparties entre des copies des modules de logiciels chargees multiples pour une utilisation dans ledit systeme de logiciel, dans lequel lesdites definitions de taches comprennent une ou plusieurs contraintes sur une ou plusieurs variables et le gestionnaire d'exception est agence pour fournir en sortie une exception lorsqu'une variable s'oppose a une contrainte pertinente pendant l'utilisation du systeme.
18. Gestionnaire d'exception destine a une utilisation dans le dispositif selon l'une quelconque des revendications 1 a 8, dans lequel les donnees acquises comprennent des donnees de programmation pour les taches, en definissant des ressources et des contraintes de temps en rapport avec la tache, et le gestionnaire d'exception est agence pour fournir en sortie une exception lorsque l'allocation de taches ne peut pas rencontrer la ressource et les contraintes de temps entre les modules de logiciels.